DA2 Case Study - Group 8

Nicholas Kappel (466328), Gwanyong Lim (465480),
Sebastian Deisel (394951), Edward Livengood (465859)

9^{th} July 2019

1 Introduction

To solve a function approximation and multi-objective optimization problem, a custom algorithm was developed to determine a set of optimal points. A process of optimal model selection, integrated with a multi-objective evolutionary algorithm was implemented in an attempt to efficiently minimize two functions within a bounded search space. This case study outlines the process and various configurations of this custom algorithm and documents its results in an attempt to produce an optimal set of points that minimizes two unknown functions.

2 Exploring the Problem

This case study deals with a function approximation and multi-objective optimization problem. Output from two unknown functions that need to be minimized simultaneously is provided. Each function consists of three inputs and one output, and is restricted to the unit hyper cube within a range of $[-5, 5]^3$. The first step was to create surrogate models to approximate each function using appropriate machine learning algorithms. The second step was to optimize both functions by finding an approximate Pareto front that minimizes both models. The end goal of this problem was to provide a set of optimal model predicted points. There were also several constraints that affect the potential solutions.

The first constraint is the amount of data available. Data is obtained by feeding three input variables (X, Y, Z) from the hyper cube search space of

$[-5, 5]^3$ into an API, which processes them through unknown functions, and returns output values. The API limits requests to 2000 points in total, suggesting that the number of points needs to be somehow distributed between both functions.

Distributing requests between the two functions could be achieved through several approaches. Points could be evenly allocated, however, this may not be the optimal approach since the functions are unknown and their complexity may also differ. Therefore, it is potentially beneficial to request more data points for a complex function and less for a simple function.

Deciding which points to request is another constraint that needs to be addressed. Requesting random points within the search space is a simple approach, but is unlikely to be the optimal way to obtain data that will create a good model. Additionally, the search space can be spanned uniformly in order to get a full picture of how each function acts along each dimension, but this could also lead to data points being requested that do not help to approximate the functions any better and waste valuable requests. To select points within the search space that will improve the model, a solution that focuses on requesting data points from important regions is required.

3 The Custom Algorithm

3.1 Selecting the Initial Data Set

For the first part of the problem the following approach was developed. To get an initial overview of how each function looked within the search space, the grid was spanned from -4.5 to 4.5 in steps of 3, resulting in 64 points from each function. However, after running more experiments on a test function API, this approach was later revised and the search space was uniformly spanned with a length of 7 from -5 to 5 in every dimension resulting in 343 points. Due to the constrained number of fetches, these two data sets were combined to form the initial data set consisting of 407 points for each function as shown in Figure 1.

A benchmark of 1,787 trained models was then run to explore which algorithms and hyperparameters have optimal performance on the initial data. The results were used to train the two models to find a Pareto front. The NSGA-II algorithm was chosen for optimization, which will be discussed later in detail. New data points to fetch were chosen from the Pareto set of each

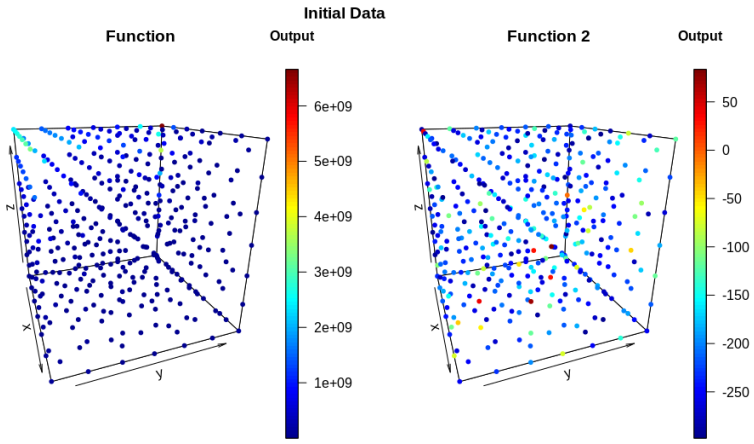

Figure 1: Visualization of the initial data set.

iteration. To avoid fetching unnecessary points, a redundancy filter based on euclidean distance with a threshold of $d(p, q) <= 0.1$ was introduced. The new data points were then used to feed back into the models. This process is then repeated until the request limit is reached. Listing 1 outlines the aforementioned process in pseudo-code.

```
data_fct_1 = fetch_init_data()
data_fct_2 = fetch_init_data()

params_model_1 = benchmark_models(data_fct_1)
params_model_2 = benchmark_models(data_fct_2)

while(data_can_be_fetched) {
    model_fct_1 = train(data_fct_1, params_model_1)
    model_fct_2 = train(data_fct_2, params_model_2)

    pareto_front = get_pareto_front(model_fct_1, model_fct_2)
    pareto_set = derive_pareto_set(Pareto_front)

    data_fct_1.append(fetch_data_fct_1(pareto_set))
    data_fct_2.append(fetch_data_fct_2(pareto_set))
}
params_model_1 = benchmark_models(data_fct_1)
params_model_2 = benchmark_models(data_fct_2)

model_fct_1 = train(data_fct_1, params_model_1)
model_fct_2 = train(data_fct_2, params_model_2)

final_pareto_front = get_pareto_front(model_fct_1, model_fct_2)
return final_pareto_front
```

Listing 1: Approach pseudocode

3.2 Model Benchmarking

From the Machine Learning in R (MLR) package, a variety of applicable regression algorithms were selected in order to benchmark their performance. This benchmark included comparing different algorithms as well as different settings for the hyperparameters of the respective algorithms.

Instead of using all regression learners available in MLR, a few learners representing different classes of algorithms were selected. The algorithms compared were: "regr.ksvm" (Support Vector Machines), "regr.randomForest" (Random Forest), "regr.nnet" (Feed-forward Neural Networks with a single hidden layer), "regr.gausspr" (Gaussian Processes) and "regr.rpart" (Decision Tree). All of these algorithms were fit several times with different combinations of hyperparameters. These combinations were the result of the Cartesian product of several hyperparameters for the respective algorithm. In total, 1,691 different models were fit. The `tuneParams()` function integrated into MLR for benchmarking is not compatible with all possible hyperparameters for specific machine learning algorithms. This limitation resulted in the exclusion of hyperparameter value combinations which would mutually exclude each other.

Benchmarking was also performed with the Keras library using 96 different configurations of feed forward neural networks. Every model that was trained for this benchmark consisted of two hidden layers. ReLu (Rectified Linear Units) was used throughout as the activation function for both hidden layers. Spanning the benchmark across a wider set of configurations including different activation functions or number of layers was not feasible for our work since such a benchmark of hundreds or thousands of combinations might have taken several days to execute. The hard coded amount of layers and activation function was therefore based on preliminary observations. All benchmarks were conducted on our initial data fetch.

After fitting all 1,787 (1,691+96) models, they were evaluated by the MSE yielded. The data used for this validation was a fixed share of 20% of the available data where for each model the validation data set was identical. No cross validation was performed for this evaluation because the R bindings of Keras don't support this feature out of the box. The benchmark for the initial data set yielded a set of hyperparameters having the lowest MSE.

Subsequently, these two configurations were used for the function approximation of our data exploration task.

Function 1	Function 2
regr.ksvm (MLR)	Keras (Feedforward)
C: 2.5	Layers: 2
Sigma: 1	Actv. fun.: ReLu
Epsilon: 0.01	Units per layer: 128
Tolerance: 0.0005	Optimizer: Adamax
	Learning rate: 0.05

Table 1: Optimal configurations for initial data sets based on the benchmark

3.3 Multi-Objective Optimization

For this case study, two evolutionary algorithms were considered: Non-dominated Sorting Genetic Algorithm (NSGA-II) and S-Metric Selection-EMOA (SMS-EMOA). NSGA-II performs well for bi-objective optimization [**Deb**, **K**]. SMS-EMOA is effective, however, it exploits Hypervolume (HV) calculation as a secondary criterion, which requires expensive computational efforts [**Beume**, **N**]. Finally, NSGA-II was chosen since the task has two objectives to optimize and avoids high computational efforts.

3.4 NSGA-II Parameter Selection

NSGA-II was implemented using the R (ecr) package. For the mutation operators, "mutPolynominal", "mutGauss", and "mutInversion" were considered; and for the recombination operators, "recSBX", "recUnifCrossover", and "recIntermediate", "recCrossover" were considered. Some operators include configurable hyperparameters such as probability parameter (p), standard deviation of the Gaussian mutation (sdev) and distance parameter of the distribution (eta). A total of 1,254 combinations were created from the result of the Cartesian product of operators and their configurable parameters. The benchmark was trained using all combinations on a test API trained model. The different NSGA-II models find each Pareto optimal front based on the trained model and is evaluated by the HV. A reference point for calculating the HV is set as the maximum value of the respective objective.

As shown in Table 2, the difference between maximum and minimum HVs is approximately 1%. Consequently, the different hyperparameters of NSGA-II do not have a strong influence on the result. Finally, the hyperparameters for NSGA-II which produced the largest HV were chosen.

	Mutation parameter	Recombination parameter	Operator combination	HV
MAX	p = 0.2, eta = 20	p = 1 eta = 5	mutPolynomial + recSBX	16399999
MIN	p = 0.5, sdev = 0.05	p = 1	mutGauss +recUnifCrossove	16228853

Table 2: Comparison of different NSGA-II models

3.5 Selecting Model Based Data Points

The limited amount of available data affects the accuracy of the model, and therefore the accuracy of NSGA-II. To identify an accurate model based Pareto front without having any knowledge about the true Pareto front, an approach was developed based on findings from the test function API. Using the test API, the Pareto front from the model output, from the actual data fetched, and from a large sample were compared. The large sample consisted of 132,651 evenly distributed data points to approximate the "true" Pareto front as close as possible. The different Pareto fronts were compared using the IGD (Inverted Generational Distance). A high correlation was observed on the IGD between the large sample and the model predictions, as well as on the IGD between the fetched data and the model predictions, as shown in Figure 2.

Tests with several hyperparameter configurations were ran and the two IGDs were recorded for each iteration of the custom algorithm. The hyperparameter configurations had little effect on the results, and a correlation greater than 0.8 was observed for nearly every run, showing that the configuration has little effect on the stability. Furthermore, this correlation was constantly greater than 0.5 even if the data set used for training consisted of only a few hundred observations per function. In the later stages of the custom algorithm where more data was available for training, the correlation increased to values around 0.9. There is uncertainty related to the interpretation of this correlation, but there was enough confidence to use this insight to select the model based data points for submission. Since there is no way to observe the true Pareto front of the problem, the following assumption was made: *The high correlation between model-to-truth IGD and model-to-fetched IGD of the test API will also be present in the real API.*

Following this assumption, a decision could be made regarding the prox-

Figure 2: IGD between model and truth as well as model and fetched data while running the custom algorithm over the test API (correlation = 0.91)

imity of the model based Pareto fronts to the non dominated points of the data fetched from the API. If the model predicted this Pareto front well, it became eligible to select the data points for submission, especially since the approach for selecting data points was based on data directly from the Pareto front we received running our algorithm.

A potential issue was that the quality of the Pareto front model predictions had a large variance as shown on Figure 2. A valid explanation of this phenomenon was never reached, and is therefore a strong indicator that applying NSGA-II on model based predictions should be further explored before being applied to future problems. Observing the correlation of the different IGDs helped to overcome this issue, but the random nature of the NSGA-II outcome is undesirable.

4 Final Assessment

4.1 Executing the Custom Algorithm

Using the benchmark hyperparameters, the custom algorithm was applied to the initial data set of 407 observations per function until the 2000 data point

limit was reached. During execution, both the model predicted Pareto front and the Pareto front of the actual data was recorded. Furthermore, the MSE for each iteration was recorded. The limit was reached after 51 iterations of the custom algorithm, resulting in an average of 11.63 observations fetched per function per iteration. After execution, IGDs between Pareto fronts from the model and the collected data were compared for each iteration. As expected, this IGD fluctuated strongly over time. Following the assumptions outlined earlier, data points which belonged to an iteration where this IGD was low compared to other iterations were selected for submission. Preference was given to later iterations due to initial observations that along the test API the correlation of the compared IGDs increased with more iterations. The optimal points selected originate from iteration 46 of the custom algorithm, meaning that the underlying surrogate model for this outcome was not trained using the entire volume of API calls. While this seems to be counter-intuitive, the correlations between the different IGDs observed for the test API justify this selection of data points. Since the IGD calculated by using the models trained on the full set of data was higher than the IGD of the selected data, the choice made for selecting these values is justified. This resulted in the 22 optimal points displayed in Table 3, which is composed of the model predicted Pareto front with the lowest IGD. Following execution of the custom algorithm, another benchmark was performed on the final data set, but failed to yield significant improvements. A final execution of the NSGA-II algorithm also failed to yield better results. Therefore, this benchmark was not taken into consideration for the final results.

4.2 Discussion of Results

Several issues were encountered during the execution of the custom algorithm. Two or more runs of an NSGA-II configuration on identical surrogate models produced very different outcomes. While no deeper investigation of this issue was performed, it was assumed that the fluctuations of the IGD could be attributed to this behavior. Another issue was the accuracy of the surrogate models used for making predictions. The MSE for the second function improved with more data being available for training the model per iteration, however, the MSE of the first function after training with all available data was slightly worse than the initially trained model. The initial assumption was that the accuracy of the models improve while fetching more data to increase the quality of predictions with each iterations; however, this

x	y	z	function1	function2
-0.624	2.733	-0.812	-2708825.000	-121.012
0.472	-0.010	4.571	509956577.769	-375.042
0.401	0.482	3.132	165742485.362	-358.887
0.474	0.275	4.368	409609027.680	-362.208
0.402	0.804	3.037	129126893.950	-352.904
0.434	0.824	-2.098	11747530.978	-349.982
0.472	0.269	4.630	470957459.375	-374.018
0.520	0.269	4.378	413629243.558	-364.639
0.394	0.092	4.437	456573577.693	-364.650
0.401	0.689	3.157	150705149.432	-358.108
0.477	0.971	-4.028	3942339.453	-322.131
0.576	4.434	2.291	-835846.767	-269.290
0.472	0.269	4.378	412977956.303	-362.627
0.402	0.546	-3.395	6982173.005	-324.803
0.403	0.597	-3.281	7568692.449	-329.678
0.409	0.516	-2.860	10168199.430	-333.386
-0.871	0.656	-4.373	-1487547.853	-183.728
-0.624	2.502	-0.812	-2458111.495	-143.711
0.433	0.509	-2.843	10363075.993	-336.886
0.432	0.945	-2.115	11540961.305	-339.727
0.404	0.723	-2.481	10944220.375	-339.320
0.411	0.591	-3.347	7213336.401	-328.902

Table 3: Input vectors selected for submission and their predicted outputs.

assumption did not hold true.

The custom algorithm was observed to explore areas of the search space that resulted in relatively low outputs for both functions. Since the predicted Pareto front is fetched after each iteration, this behavior was expected. Figure 3 displays the algorithms tendency to focus on fetching points that result in relatively low output values for both functions.

4.3 Outlook & Conclusion

The custom algorithm and approach developed for this case study has much room for improvement. Firstly, the effect of the NSGA-II algorithms random nature should be better understood. While attempts to explain the

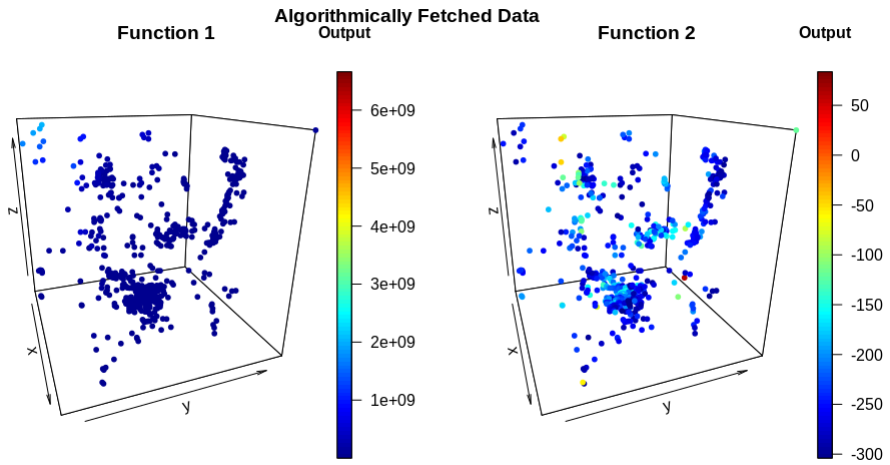

Figure 3: Visualization of data fetched by the custom algorithm.

fluctuation of IGD values in different iterations were made, no conclusion was reached. Additionally, IGD values were only calculated between model predictions and the Pareto front of the actual data at each iteration. In retrospect, this is likely not an optimal approach, and comparing against the Pareto front of the final data set may have yielded more accurate and interpretable results. The algorithm should also be tested on larger data sets to explore its scalability.

While the custom algorithm developed needs to be improved upon, it still has potential and interesting implications. For multi-objective optimization problems, this approach performs well at exploring important data regions. Since the true Pareto set of a multi-objective optimization problem is often unknown, further development of this approach could potentially yield promising approximations.

References

[Beume, N] Beume, N., Naujoks, B., and Emmerich, M. 2007. "SMS-EMOA: Multiobjective Selection Based on Dominated Hypervolume," European Journal of Operational Research (181:3), pp. 1653–1669. (https://doi.org/10.1016/j.ejor.2006.08.008).

[Deb, K] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. 2002. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation (6:2), pp. 182–197. (https://doi.org/10.1109/4235.996017).